

**C13**

```

libdebugon
  daemon_daemon.....10 (EMXD)spatch.c)
  daemon_cleanup.....15 (EMXD)spatch.c)
  daemon_check_proper_ID.....7 (EMXD)spatch.c)
  daemon_cleanup.....16 (EMXD)spatch.c)
  daemon_initialize_logging.....9 (EMXD)spatch.c)
  daemon_specific_initialization.....4 (EMXD)spatch.c)
  clear_log_usage.....3 (EMXD)spatch.c)
  X11_initialize.....3 (EMXD)spatch.c)
  parse_commandline.....12 (EMXD)spatch.c)
  rpc_init.....12 (EMXD)spatch.c)
  rpc_run.....15 (EMXD)spatch.c)

```



```

1  /*
2  */
3  /* Copyright 1996,1997 BMC Corporation
4
5  */
6  /* EDMDspatch.c
7
8  */
9  /* Mission Statement: this is the main service file for the dispatch
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666

```

Page 3 of 20	kill_handler	Thu Dec 27 11:23:05 2007
99	/*..... 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200	<pre> /*..... ** Routine: kill_handler ** Input: int signal - the signal which was received. ** Output: Will log messages calling what action is being taken. ** Return Codes: **   exits with the number of the signal received ** Purpose: This routine handles specific signals i.e. SIGINT,             SIGQUIT, SIGTERM. Each results in a log entry and an exit. **             Intended caller: Internal only.             ***** */ static void kill_handler (in int signal) {     struct timeval t;     time_t current_time;     char *buff;      /* If main exits, it calls this routine with signal 0 */      /* Unregister the interface */     (void) csc_unregister_server_interface(&amp;t_spec, &amp;status);      /* If the unregister fails, report the problem, but continue */     if ( status != error_status_OK )     {         buff = (char *) csc_get_error( status );         (void) EDMDispatch_logent(             FILE_, LINE_, LOG_ERR, MESSAGE_NO_LOGIN, 0,             "CSC_SERVER_LOGIN FAILED: %s", buff );         status, (buff ? buff : "Unknown error" );     }      /* Get the current time */     (void) time(&amp;current_time);     ctimbuf = ctimex(current_time);      /* Overlay newline with null - buff should always be 26 bytes long */     ctimbuf[ strlen(ctimbuf) - 1 ] = 0;      (void) EDMDispatch_logent(         FILE_, LINE_, LOG_INFO, MESSAGE_SHUTDOWN, 0,         "Shutting down at %s due to signal %d", ctimbuf,         signal);      /* Remove our lock file.     **     ** (void) BaddestCopyFile(PIDDIRM);     exit(signal);     */ } */ End of kill_handler() EDMDispatch.c Thu Dec 27 11:23:05 2007 </pre>
Page 4 of 20	display_usage	Thu Dec 27 11:23:05 2007
159	160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200	<pre> /*..... ** Function Name: **   display_usage ** Simply displays the usage ** ** Call Arguments: **   Program name ** Error Outputs and Side Effects: **   Prints usage. **   Special Considerations: **   None.             ***** */ static void display_usage (in char *programme) {     /* Print out usage stmt. */      fprintf(stderr, "Usage: %s [-d]\n", programme);     fprintf ( stderr, "-d keep the daemon from forking so debugging is easier\n");     }     /* end display_usage () */ } </pre>
Page 3 of 20	EDMDispatch.c 3	Thu Dec 27 11:23:05 2007
Page 4 of 20	EDMDispatch.c 4	Thu Dec 27 11:23:05 2007

```

187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

/*
** Routine: daemon_catch_interrupts
** Inputs: None
** Outputs: None
** Return Codes: None
** Purpose: Set up signals for service. On NT we will have to
** catch up signals that we are catching with.
** In this case we are catching SIGTERM, SIGINT, and
** SIGQUIT and ignoring anything else.
** Intended caller: Internal only.
**
*****
*/
void daemon_catch_interrupts()
{
    struct sigaction sactions;

    /* Signal actions */
    ZENO( sactions );

    /* Set an empty list so we can set signals we want to handle
    (void) sigemptyset( &sactions.sa_mask );

    /* Add signals that we want to handle
    (void) sigaddset( &sactions.sa_mask, SIGTERM );
    (void) sigaddset( &sactions.sa_mask, SIGINT );
    (void) sigaddset( &sactions.sa_mask, SIGQUIT );

    /* Setup the signal handler */
    sactions.sa_handler = kill_handler;

    /* Assign handler to each signal we are interested in.
    (void) sigaction( SIGTERM, &sactions, NULL );
    (void) sigaction( SIGINT, &sactions, NULL );
    (void) sigaction( SIGQUIT, &sactions, NULL );

    /* Setup mask so we can specify what signals we will ignore.
    (void) sigfillset( &sactions.sa_mask );

    /* We want to ignore everything except those we have set up
    above so remove those from the list.
    (void) sigdelset( &sactions.sa_mask, SIGTERM );
    (void) sigdelset( &sactions.sa_mask, SIGINT );
    (void) sigdelset( &sactions.sa_mask, SIGQUIT );
}
*/

```

```

249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

/* Set the mask. Since no other threads have been started,
all threads will get this mask.
(void) thr_sigsetmask( SIG_SETMASK, &sa_mask, NULL );
}
*/

```

```

226 */
227 ** Routine: daemon_check_proper_ID
228 **
229 ** Inputs:  None
230 **
231 ** Outputs:  None
232 **
233 ** Return Codes:
234 **   exits with an error when the user is not root
235 **
236 ** Purpose:  Checks user's ID and determines if the user is allowed
237 **           to execute services. If there are no constraints then this
238 **           function may be blank.
239 **
240 ** Intended caller:  Internal only.
241 **
242 .....
243 */
244 void daemon_check_proper_ID()
245 {
246     /*
247     ** Check for root
248     */
249     if (geteuid() != E_ROOTUID)
250     {
251         (void) EMDDispatch_logent(
252             FILE, LINE, LOG_ERR, DAEMON_NOTSUPERUSER, 0,
253             "True the run as superuser, uid was %d.",
254             geteuid());
255     }
256 }
257
258
259
260

```

```

291 */
292 ** Routine: parse_commandline
293 **
294 ** Inputs:  argc, argv (command line arguments)
295 **
296 ** Outputs:  None
297 **
298 ** Return Codes:
299 **   exits with an error when the user types a bad argument
300 **
301 ** Purpose:  Parses command line arguments and sets flags. If there
302 **           are no flags to be set then this function may be empty.
303 **
304 ** Intended caller:  Internal only.
305 **
306 .....
307 */
308 void parse_commandline(int argc, char *argv[])
309 {
310     int opt; /* Process options */
311     int i;
312     while ((opt = getopt(argc, argv, "d?")) != EOF)
313     {
314         commandlinearg = argv;
315         switch(opt)
316         {
317             case 'd':
318                 case 'd':
319                     case 'd':
320                     case 'd':
321                     case 'd':
322                     case 'd':
323                     case 'd':
324                     case 'd':
325                     case 'd':
326                     case 'd':
327                     case 'd':
328                     case 'd':
329                     case 'd':
330                     case 'd':
331                     case 'd':
332                     case 'd':
333                     case 'd':
334                     case 'd':
335                     case 'd':
336                     case 'd':
337                     case 'd':
338                     case 'd':
339                     case 'd':
340                     case 'd':
341                     case 'd':
342                     case 'd':
343                     case 'd':
344                     case 'd':
345                     case 'd':
346                     case 'd':
347                     case 'd':
348                     case 'd':
349                     case 'd':
350                     case 'd':
351                     case 'd':
352                     case 'd':
353                     case 'd':
354                     case 'd':
355                     case 'd':
356                     case 'd':
357                     case 'd':
358                     case 'd':
359                     case 'd':
360                     case 'd':
361                     case 'd':
362                     case 'd':
363                     case 'd':
364                     case 'd':
365                     case 'd':
366                     case 'd':
367                     case 'd':
368                     case 'd':
369                     case 'd':
370                     case 'd':
371                     case 'd':
372                     case 'd':
373                     case 'd':
374                     case 'd':
375                     case 'd':
376                     case 'd':
377                     case 'd':
378                     case 'd':
379                     case 'd':
380                     case 'd':
381                     case 'd':
382                     case 'd':
383                     case 'd':
384                     case 'd':
385                     case 'd':
386                     case 'd':
387                     case 'd':
388                     case 'd':
389                     case 'd':
390                     case 'd':
391                     case 'd':
392                     case 'd':
393                     case 'd':
394                     case 'd':
395                     case 'd':
396                     case 'd':
397                     case 'd':
398                     case 'd':
399                     case 'd':
400                     case 'd':
401                     case 'd':
402                     case 'd':
403                     case 'd':
404                     case 'd':
405                     case 'd':
406                     case 'd':
407                     case 'd':
408                     case 'd':
409                     case 'd':
410                     case 'd':
411                     case 'd':
412                     case 'd':
413                     case 'd':
414                     case 'd':
415                     case 'd':
416                     case 'd':
417                     case 'd':
418                     case 'd':
419                     case 'd':
420                     case 'd':
421                     case 'd':
422                     case 'd':
423                     case 'd':
424                     case 'd':
425                     case 'd':
426                     case 'd':
427                     case 'd':
428                     case 'd':
429                     case 'd':
430                     case 'd':
431                     case 'd':
432                     case 'd':
433                     case 'd':
434                     case 'd':
435                     case 'd':
436                     case 'd':
437                     case 'd':
438                     case 'd':
439                     case 'd':
440                     case 'd':
441                     case 'd':
442                     case 'd':
443                     case 'd':
444                     case 'd':
445                     case 'd':
446                     case 'd':
447                     case 'd':
448                     case 'd':
449                     case 'd':
450                     case 'd':
451                     case 'd':
452                     case 'd':
453                     case 'd':
454                     case 'd':
455                     case 'd':
456                     case 'd':
457                     case 'd':
458                     case 'd':
459                     case 'd':
460                     case 'd':
461                     case 'd':
462                     case 'd':
463                     case 'd':
464                     case 'd':
465                     case 'd':
466                     case 'd':
467                     case 'd':
468                     case 'd':
469                     case 'd':
470                     case 'd':
471                     case 'd':
472                     case 'd':
473                     case 'd':
474                     case 'd':
475                     case 'd':
476                     case 'd':
477                     case 'd':
478                     case 'd':
479                     case 'd':
480                     case 'd':
481                     case 'd':
482                     case 'd':
483                     case 'd':
484                     case 'd':
485                     case 'd':
486                     case 'd':
487                     case 'd':
488                     case 'd':
489                     case 'd':
490                     case 'd':
491                     case 'd':
492                     case 'd':
493                     case 'd':
494                     case 'd':
495                     case 'd':
496                     case 'd':
497                     case 'd':
498                     case 'd':
499                     case 'd':
500                     case 'd':
501                     case 'd':
502                     case 'd':
503                     case 'd':
504                     case 'd':
505                     case 'd':
506                     case 'd':
507                     case 'd':
508                     case 'd':
509                     case 'd':
510                     case 'd':
511                     case 'd':
512                     case 'd':
513                     case 'd':
514                     case 'd':
515                     case 'd':
516                     case 'd':
517                     case 'd':
518                     case 'd':
519                     case 'd':
520                     case 'd':
521                     case 'd':
522                     case 'd':
523                     case 'd':
524                     case 'd':
525                     case 'd':
526                     case 'd':
527                     case 'd':
528                     case 'd':
529                     case 'd':
530                     case 'd':
531                     case 'd':
532                     case 'd':
533                     case 'd':
534                     case 'd':
535                     case 'd':
536                     case 'd':
537                     case 'd':
538                     case 'd':
539                     case 'd':
540                     case 'd':
541                     case 'd':
542                     case 'd':
543                     case 'd':
544                     case 'd':
545                     case 'd':
546                     case 'd':
547                     case 'd':
548                     case 'd':
549                     case 'd':
550                     case 'd':
551                     case 'd':
552                     case 'd':
553                     case 'd':
554                     case 'd':
555                     case 'd':
556                     case 'd':
557                     case 'd':
558                     case 'd':
559                     case 'd':
560                     case 'd':
561                     case 'd':
562                     case 'd':
563                     case 'd':
564                     case 'd':
565                     case 'd':
566                     case 'd':
567                     case 'd':
568                     case 'd':
569                     case 'd':
570                     case 'd':
571                     case 'd':
572                     case 'd':
573                     case 'd':
574                     case 'd':
575                     case 'd':
576                     case 'd':
577                     case 'd':
578                     case 'd':
579                     case 'd':
580                     case 'd':
581                     case 'd':
582                     case 'd':
583                     case 'd':
584                     case 'd':
585                     case 'd':
586                     case 'd':
587                     case 'd':
588                     case 'd':
589                     case 'd':
590                     case 'd':
591                     case 'd':
592                     case 'd':
593                     case 'd':
594                     case 'd':
595                     case 'd':
596                     case 'd':
597                     case 'd':
598                     case 'd':
599                     case 'd':
600                     case 'd':
601                     case 'd':
602                     case 'd':
603                     case 'd':
604                     case 'd':
605                     case 'd':
606                     case 'd':
607                     case 'd':
608                     case 'd':
609                     case 'd':
610                     case 'd':
611                     case 'd':
612                     case 'd':
613                     case 'd':
614                     case 'd':
615                     case 'd':
616                     case 'd':
617                     case 'd':
618                     case 'd':
619                     case 'd':
620                     case 'd':
621                     case 'd':
622                     case 'd':
623                     case 'd':
624                     case 'd':
625                     case 'd':
626                     case 'd':
627                     case 'd':
628                     case 'd':
629                     case 'd':
630                     case 'd':
631                     case 'd':
632                     case 'd':
633                     case 'd':
634                     case 'd':
635                     case 'd':
636                     case 'd':
637                     case 'd':
638                     case 'd':
639                     case 'd':
640                     case 'd':
641                     case 'd':
642                     case 'd':
643                     case 'd':
644                     case 'd':
645                     case 'd':
646                     case 'd':
647                     case 'd':
648                     case 'd':
649                     case 'd':
650                     case 'd':
651                     case 'd':
652                     case 'd':
653                     case 'd':
654                     case 'd':
655                     case 'd':
656                     case 'd':
657                     case 'd':
658                     case 'd':
659                     case 'd':
660                     case 'd':
661                     case 'd':
662                     case 'd':
663                     case 'd':
664                     case 'd':
665                     case 'd':
666                     case 'd':
667                     case 'd':
668                     case 'd':
669                     case 'd':
670                     case 'd':
671                     case 'd':
672                     case 'd':
673                     case 'd':
674                     case 'd':
675                     case 'd':
676                     case 'd':
677                     case 'd':
678                     case 'd':
679                     case 'd':
680                     case 'd':
681                     case 'd':
682                     case 'd':
683                     case 'd':
684                     case 'd':
685                     case 'd':
686                     case 'd':
687                     case 'd':
688                     case 'd':
689                     case 'd':
690                     case 'd':
691                     case 'd':
692                     case 'd':
693                     case 'd':
694                     case 'd':
695                     case 'd':
696                     case 'd':
697                     case 'd':
698                     case 'd':
699                     case 'd':
700                     case 'd':
701                     case 'd':
702                     case 'd':
703                     case 'd':
704                     case 'd':
705                     case 'd':
706                     case 'd':
707                     case 'd':
708                     case 'd':
709                     case 'd':
710                     case 'd':
711                     case 'd':
712                     case 'd':
713                     case 'd':
714                     case 'd':
715                     case 'd':
716                     case 'd':
717                     case 'd':
718                     case 'd':
719                     case 'd':
720                     case 'd':
721                     case 'd':
722                     case 'd':
723                     case 'd':
724                     case 'd':
725                     case 'd':
726                     case 'd':
727                     case 'd':
728                     case 'd':
729                     case 'd':
730                     case 'd':
731                     case 'd':
732                     case 'd':
733                     case 'd':
734                     case 'd':
735                     case 'd':
736                     case 'd':
737                     case 'd':
738                     case 'd':
739                     case 'd':
740                     case 'd':
741                     case 'd':
742                     case 'd':
743                     case 'd':
744                     case 'd':
745                     case 'd':
746                     case 'd':
747                     case 'd':
748                     case 'd':
749                     case 'd':
750                     case 'd':
751                     case 'd':
752                     case 'd':
753                     case 'd':
754                     case 'd':
755                     case 'd':
756                     case 'd':
757                     case 'd':
758                     case 'd':
759                     case 'd':
760                     case 'd':
761                     case 'd':
762                     case 'd':
763                     case 'd':
764                     case 'd':
765                     case 'd':
766                     case 'd':
767                     case 'd':
768                     case 'd':
769                     case 'd':
770                     case 'd':
771                     case 'd':
772                     case 'd':
773                     case 'd':
774                     case 'd':
775                     case 'd':
776                     case 'd':
777                     case 'd':
778                     case 'd':
779                     case 'd':
780                     case 'd':
781                     case 'd':
782                     case 'd':
783                     case 'd':
784                     case 'd':
785                     case 'd':
786                     case 'd':
787                     case 'd':
788                     case 'd':
789                     case 'd':
790                     case 'd':
791                     case 'd':
792                     case 'd':
793                     case 'd':
794                     case 'd':
795                     case 'd':
796                     case 'd':
797                     case 'd':
798                     case 'd':
799                     case 'd':
800                     case 'd':
801                     case 'd':
802                     case 'd':
803                     case 'd':
804                     case 'd':
805                     case 'd':
806                     case 'd':
807                     case 'd':
808                     case 'd':
809                     case 'd':
810                     case 'd':
811                     case 'd':
812                     case 'd':
813                     case 'd':
814                     case 'd':
815                     case 'd':
816                     case 'd':
817                     case 'd':
818                     case 'd':
819                     case 'd':
820                     case 'd':
821                     case 'd':
822                     case 'd':
823                     case 'd':
824                     case 'd':
825                     case 'd':
826                     case 'd':
827                     case 'd':
828                     case 'd':
829                     case 'd':
830                     case 'd':
831                     case 'd':
832                     case 'd':
833                     case 'd':
834                     case 'd':
835                     case 'd':
836                     case 'd':
837                     case 'd':
838                     case 'd':
839                     case 'd':
840                     case 'd':
841                     case 'd':
842                     case 'd':
843                     case 'd':
844                     case 'd':
845                     case 'd':
846                     case 'd':
847                     case 'd':
848                     case 'd':
849                     case 'd':
850                     case 'd':
851                     case 'd':
852                     case 'd':
853                     case 'd':
854                     case 'd':
855                     case 'd':
856                     case 'd':
857                     case 'd':
858                     case 'd':
859                     case 'd':
860                     case 'd':
861                     case 'd':
862                     case 'd':
863                     case 'd':
864                     case 'd':
865                     case 'd':
866                     case 'd':
867                     case 'd':
868                     case 'd':
869                     case 'd':
870                     case 'd':
871                     case 'd':
872                     case 'd':
873                     case 'd':
874                     case 'd':
875                     case 'd':
876                     case 'd':
877                     case 'd':
878                     case 'd':
879                     case 'd':
880                     case 'd':
881                     case 'd':
882                     case 'd':
883                     case 'd':
884                     case 'd':
885                     case 'd':
886                     case 'd':
887                     case 'd':
888                     case 'd':
889                     case 'd':
890                     case 'd':
891                     case 'd':
892                     case 'd':
893                     case 'd':
894                     case 'd':
895                     case 'd':
896                     case 'd':
897                     case 'd':
898                     case 'd':
899                     case 'd':
900                     case 'd':
901                     case 'd':
902                     case 'd':
903                     case 'd':
904                     case 'd':
905                     case 'd':
906                     case 'd':
907                     case 'd':
908                     case 'd':
909                     case 'd':
910                     case 'd':
911                     case 'd':
912                     case 'd':
913                     case 'd':
914                     case 'd':
915                     case 'd':
916                     case 'd':
917                     case 'd':
918                     case 'd':
919                     case 'd':
920                     case 'd':
921                     case 'd':
922                     case 'd':
923                     case 'd':
924                     case 'd':
925                     case 'd':
926                     case 'd':
927                     case 'd':
928                     case 'd':
929                     case 'd':
930                     case 'd':
931                     case 'd':
932                     case 'd':
933                     case 'd':
934                     case 'd':
935                     case 'd':
936                     case 'd':
937                     case 'd':
938                     case 'd':
939                     case 'd':
940                     case 'd':
941                     case 'd':
942                     case 'd':
943                     case 'd':
944                     case 'd':
945                     case 'd':
946                     case 'd':
947                     case 'd':
948                     case 'd':
949                     case 'd':
950                     case 'd':
951                     case 'd':
952                     case 'd':
953                     case 'd':
954                     case 'd':
955                     case 'd':
956                     case 'd':
957                     case 'd':
958                     case 'd':
959                     case 'd':
960                     case 'd':
961                     case 'd':
962                     case 'd':
963                     case 'd':
964                     case 'd':
965                     case 'd':
966                     case 'd':
967                     case 'd':
968                     case 'd':
969                     case 'd':
970                     case 'd':
971                     case 'd':
972                     case 'd':
973                     case 'd':
974                     case 'd':
975                     case 'd':
976                     case 'd':
977                     case 'd':
978                     case 'd':
979                     case 'd':
980                     case 'd':
981                     case 'd':
982                     case 'd':
983                     case 'd':
984                     case 'd':
985                     case 'd':
986                     case 'd':
987                     case 'd':
988                     case 'd':
989                     case 'd':
990                     case 'd':
991                     case 'd':
992                     case 'd':
993                     case 'd':
994                     case 'd':
995                     case 'd':
996                     case 'd':
997                     case 'd':
998                     case 'd':
999                     case 'd':
1000                    case 'd':

```

```

332 //.....
333 **
334 ** Routine: daemon_initialize_logging
335 ** Inputs: None
336 ** Outputs: None
337 ** Return Codes: None
338 ** Purpose: Do whatever it takes to initialize logging. In the near
339             future this may involve doing something with catalogs or
340             calling higher level logging functions which encapsulate
341             these things.
342 **
343 **
344 **
345 **
346 **
347 ** Intended caller: Internal only.
348 **
349 **
350 **
351 **
352 **
353 **
354 **
355 **
356 **
357 **
358 **
359 **
360 **
361 **
362 **
363 **
364 **
365 **
366 **
367 **
368 **
369 **
370 **
371 **
372 **
373 **
374 **
375 **
376 **
377 **
378 **
379 **
380 **
381 **
382 **
383 **
384 **
385 **
386 **
387 **
388 **
389 **
390 **
391 **
392 **
393 **
394 **
395 **
396 **
397 **
398 **
399 **
400 **
401 **
402 **
403 **
404 **
405 **
406 **
407 **
408 **
409 **
410 **
411 **
412 **
413 **
414 **
415 **
416 **
417 **
418 **

```

```

360 //.....
361 **
362 ** Routine: daemon_become_daemon
363 ** Inputs: None
364 ** Outputs: None
365 ** Return Codes:
366             exits with an error code if initialization fails
367 ** Purpose: This function is for doing the forking etc. under UNIX.
368             It is unknown what will be necessary under NT.
369 ** Intended caller: Internal only.
370 **
371 **
372 **
373 **
374 **
375 **
376 **
377 **
378 **
379 **
380 **
381 **
382 **
383 **
384 **
385 **
386 **
387 **
388 **
389 **
390 **
391 **
392 **
393 **
394 **
395 **
396 **
397 **
398 **
399 **
400 **
401 **
402 **
403 **
404 **
405 **
406 **
407 **
408 **
409 **
410 **
411 **
412 **
413 **
414 **
415 **
416 **
417 **
418 **

```





```

483 3         (void) EMDDispatch_logent( FILE, LINE, LOG_ERR,
484 3             MESSAGE_NO_LOGIN, 0,
485 3             "CSC_SERVER_LOGIN failed: %s",
486 3             status, {
487 3                 ebuff ? ebuff : "Unknown error" );
488 2     }
489 2     /* If the failure was due to unavailable client,
490 2     * pause and then try again.
491 2     */
492 2     if ( status == SEC_SRV_SERVER_UNAVAILABLE )
493 2     {
494 2         /*
495 2          * uses sleep when SIGRRC, otherwise uses
496 2          * pthread call to delay for the specified
497 2          * time
498 2          */
499 2         CSC_SLEEP(sleep_interval);
500 2         continue;
501 2     }
502 2
503 2     /* If we get here, we had a unexpected failure. */
504 2     (void) EMDDispatch_logent( FILE, LINE, LOG_ERR,
505 2         MESSAGE_NO_LOGIN, 0,
506 2         "The service cannot log in as
507 2         required");
508 2
509 2     exit(1);
510 2
511 2     }
512 2     /* We need to initialize the authorization module before we
513 2     * do
514 2     */
515 2     /* a listen.
516 2     (void)csc_authorization_init(&status);
517 2
518 2     if ( status != error_status_ok )
519 2     {
520 2         ebuff = (char *) csc_get_error( status );
521 2         (void) EMDDispatch_logent( FILE, LINE, LOG_ERR,
522 2             MESSAGE_MONAUTHORIZATION, 0,
523 2             "CSC_AUTHORIZATION_INIT failed: %s",
524 2             status, {
525 2                 ebuff ? ebuff : "Unknown error" );
526 2         exit(1);
527 2     }
528 2
529 2     (void) csc_register_server_interface( &Espec,
530 2         SERVER_INTERFACE,
531 2         &status );
532 2
533 2     if ( status != error_status_ok )
534 2     {
535 2         ebuff = (char *) csc_get_error( status );
536 2         (void) EMDDispatch_logent( FILE, LINE, LOG_ERR,
537 2             MESSAGE_CANTREGISTER, 0,
538 2             "CSC_REGISTER_SERVER_INTERFACE failed:
539 2             %s",
540 2             status, {

```

```

343 2         }
344 2         exit(1);
345 2     }

```



```

638 1 getLimit(rlimit_max, &rlim);
640 1 {void} EXMDispatch_logent(
641 1     __FILE__, __LINE__, LOG_INFO, MESSAGE_STARTUP, 0,
642 1     "Service allows %d open files", rlim_max );
643 1 /* Initialize service launcher */
644 1 ret = EXMDispatchInit();
645 1 if (ret != 0)
646 1 {
647 2     (void) EXMDispatch_logent(
648 2         __FILE__, __LINE__, LOG_INFO, 0, 0,
649 2         "Service launcher failed returning %d",
650 2         ret );
651 1 }
652 1
653 1 /*
654 1  * Start the other threads in the daemon. The main thread
655 1  * becomes the RPC thread. Backgroundage is the entry point
656 1  * for the data collection thread. Backgroundcleanup is the
657 1  * entry point for the data expiration thread.
658 1  */
659 1 /* Thread create(background, NULL, DispatchDaemon, &ret, NULL); */
660 1 pthread_create(&mainid, NULL, DispatchDaemon, &ret, NULL);
661 1 pthread_create(&mainid, NULL, DispatchBackground, NULL);
662 1 rpc_init();
663 1 rpc_run();
664 1 }

```

```

666 1 /*
667 1  *
668 1  * Routine: daemon_cleanup
669 1  * Inputs:  None
670 1  * Outputs: None
671 1  * Return Codes:
672 1  *     None
673 1  * Purpose:  Call function which will clean up daemon properly.
674 1  * Intended caller: Internal only.
675 1  */
676 1
677 1
678 1
679 1
680 1
681 1
682 1
683 1
684 1
685 1
686 1
687 1
688 1

```

```

void
daemon_cleanup()
{
    kill_handler( 0 );
}

```



```
PreSessionInfo          5 (EDMD)spatchService.c)
dd_getServiceStatus_1_svc... 3 (EDMD)spatchService.c)
dd_getServiceStatus_2_svc... 4 (EDMD)spatchService.c)
dd_initialize_1_svc... 2 (EDMD)spatchService.c)
```



```

1 //
2 // Copyright 1996, 1997 EMC Corporation
3 //
4
5
6 /* EDMDispatchService.c
7
8 *
9 * Mission Statement: RPC entry points.
10 *
11 * Primary Data Action On:
12 *
13 * Compile-Time Options:
14 *
15 * Basic idea here:
16 */
17
18 #if !defined(lint)
19 static char RCS_id [] = "@(#)SRCFile: EDMDispatchService.c,v $ "
20
21 /*$Revision: 1.0 $ */
22 /*Date: 1997/02/06 20:49:15 $* :
23
24 #endif
25
26 #include <sys/c_portable.h>
27 #include <sys/inout.h>
28 #include <logging/logging.h>
29 #include <cscc/csccomm.h>
30
31 #include <restore/csc_EDMDispatch.h>
32 #include <restore/dispatch_demon.h>
33 #include <EDMDispatchLog.h>
34 #include <EDMDispatchSession.h>
35
36 //
37 // These are all the rpc entry points for the dispatch demon.
38 // The dispatch demon is multithreaded and it is the main thread
39 // which handles all incoming RPC. OMC RPC is single threaded
40 // so each call blocks other RPC calls. This provides us some
41 // safety in the way we handle our data and limits our exposure
42 // to unexpected multithreading problems.
43 //
44 static void FreeSessionInfo(SessionInfo *);
45
46 //*****
47 //
48 // Routine: dd_initialize_1
49 //
50 // Inputs: DD_initialize_args * - args for the restore initialize
51 //         call
52 //
53 // Outputs: None
54 //
55 // Return Codes:
56 //
57 // Purpose: Function to create a restore session.
58 //
59 // Intended caller: Internal only
60 //
61 //*****
62
63 DD_initialize_result *

```

```

64 dd_initialize_1_svc(
65 1 {
66 1 static DD_initialize_result argz;
67 1
68 1 InitializeSession(arg, req, kargz);
69 1
70 1 return kargz;
71 1
72 }

```



```

73  /.....
74  **
75  ** Routine: cd.getservicestatus_1
76  ** Inputs:  DD.getservicestatus_args * - args for the
77  **          getservicestatus call
78  ** Outputs: None
79  ** Return Codes:
80  **             DD.getservicestatus_result * - result of status function
81  **             call
82  **
83  ** Purpose: Function to poll for status on a session.
84  **          Intended caller: Internal Only.
85  **
86  **
87  .....
88  */
89
90  DD.getservicestatus_result *
91  cd.getservicestatus_1_svc(
92  {
93      IN DD.getservicestatus_args *arg, IN struct svc_req *req }
94  )
95  {
96      static DD.getservicestatus_result argzz;
97      GetDispatchStatus(arg, argzz);
98      return argzz;
99  }

```

```

100 /.....
101 **
102 ** Routine: cd.getsessioninfo_1
103 ** Inputs:  DD.getservicestatus_args * - args for the getsessioninfo
104 **          call
105 ** Outputs: None
106 ** Return Codes:
107 **             SessionBlock * - result of session info call
108 **             SessionBlock * - result of session info call
109 **             SessionBlock * - result of session info call
110 **             SessionBlock * - result of session info call
111 ** Purpose: Function to get information on all sessions.
112 **          Intended caller: Internal Only.
113 **
114 **
115 .....
116 */
117
118 SessionBlock *
119 cd.getsessioninfo_1_svc(
120 {
121     static SessionBlock argzz;
122     static boolean tv first = TRUE;
123     if (first)
124     {
125         memset(&argzz, 0, sizeof(argzz));
126         first = FALSE;
127     }
128     else
129     {
130         FreeSessionInfo(&argzz.sess);
131         argzz.sess = NULL;
132     }
133     GetDispatchInfo(&arg, &argzz);
134     return argzz;
135 }
136
137 }

```

```
139 /*****
140 **
141 ** Routine: FreeSessionInfo
142 **
143 ** Inputs: SessionInfo * - arg to free
144 **
145 ** Outputs: None
146 **
147 ** Return Codes:
148 **
149 **      None
150 **
151 ** Purpose: Function to free all SessionInfo structures in a list.
152 ** Intended caller: Internal Only.
153 *****/
154
155 static void FreeSessionInfo(SessionInfo *sess)
156 {
157     if (sess == NULL)
158         return;
159
160     if (sess -> next != NULL)
161         FreeSessionInfo(sess -> next);
162     free(sess);
163 }
```

